CLAIMS

What is claimed is:

1. A digital circuit with redundancy protection, the digital circuit comprising:

a time multiplexer to assign a plurality of redundant data into a plurality of time slots;

a time-multiplexed circuit coupled to the time multiplexer, the time-multiplexed circuit processing the plurality of redundant data in the plurality of time slots to generate a plurality of redundant results respectively; and

a voting circuit coupled to the time-multiplexed circuit, the voting circuit processing the plurality of redundant results to maintain data integrity.

2. A digital circuit as in claim 1, wherein the voting circuit determines an output result according to a majority of the redundant results.

3. A digital circuit as in claim 2, wherein the voting circuit identifies a faulty one of the plurality of redundant results.

4. A digital circuit as in claim 3, further comprising:

a reload logic circuit coupled to the voting circuit and the time-multiplexed circuit, the time-multiplexed circuit having a plurality of state memory elements to store a plurality of redundant states, the reload logic circuit copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

5. A digital circuit as in claim 1, wherein the voting circuit detects a computation error according to the redundant results.

6. A digital circuit as in claim 5, further comprising:

a restart logic circuit coupled to the voting circuit and the time-multiplexed circuit, the restart logic circuit causing the time-multiplexed circuit to reprocessing the plurality of redundant data in response to the voting circuit detecting the computation error.

7. A digital circuit as in claim 1, wherein the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results.

8. A digital circuit as in claim 1, wherein the time-multiplexed circuit comprises a combinatorial logic circuit.

9. A digital circuit as in claim 1, wherein the time-multiplexed circuit pipelines processing of the plurality of redundant data.

10. A digital circuit as in claim 9, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit.

11. A digital circuit as in claim 10, wherein the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

12. A digital circuit as in claim 9, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit.

13. A digital circuit as in claim 12, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel.

14. A digital circuit as in claim 1, wherein the time-multiplexed circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

15. A digital circuit as in claim 1, wherein the digital circuit is integrated on one chip.

16. A method to process data on a digital circuit with redundancy protection, the method comprising:

assigning a plurality of redundant data into a plurality of time slots;

processing the plurality of redundant data in the plurality of time slots in a time-multiplexed circuit of the digital circuit to generate a plurality of redundant results respectively; and

processing the plurality of redundant results to maintain data integrity.

17. A method as in claim 16, wherein the plurality of redundant results are processed to generate an output result according to a majority of the redundant results.

18. A method as in claim 17, wherein the plurality of redundant results are processed to identify a faulty one of the plurality of redundant results.

19. A method as in claim 18, wherein the time-multiplexed circuit has a plurality of state memory elements to store a plurality of redundant states; and the method further comprises:

copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

20. A method as in claim 16, wherein said processing the plurality of redundant results to maintain data integrity comprises detecting a computation error according to the redundant results.

21. A method as in claim 20, further comprising:

reprocessing, responsive to the computation error, the plurality of redundant data in a plurality of time slots in the time-multiplexed circuit of the digital circuit to generate a plurality of redundant results respectively.

22. A method as in claim 16, wherein the plurality of redundant results are processed to determine, in the plurality of time slots, a plurality of voting results.

23. A method as in claim 16, wherein the time-multiplexed circuit comprises a combinatorial logic circuit.

24. A method as in claim 16, wherein the time-multiplexed circuit pipelines processing of the plurality of redundant data.

25. A method as in claim 24, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit.

26. A method as in claim 25, wherein the plurality of redundant data is for a first channel; a plurality of redundant data for a second channel is assigned into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

27. A method as in claim 24, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit.

28. A method as in claim 27, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel.

29. A method as in claim 16, wherein the time-multiplexed circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

30. A method as in claim 16, wherein the digital circuit is integrated on one chip.

31. A digital circuit for data processing with redundancy protection, the digital circuit comprising:

means for assigning a plurality of redundant data into a plurality of time slots;

means for processing the plurality of redundant data in the plurality of time slots to generate a plurality of redundant results respectively; and

means for processing the plurality of redundant results to maintain data integrity.

32. A digital circuit as in claim 31, wherein the plurality of redundant results are processed to generate an output result according to a majority of the redundant results.

33. A digital circuit as in claim 32, wherein the plurality of redundant results are processed to identify a faulty one of the plurality of redundant results.

34. A digital circuit as in claim 33, wherein said means for processing the plurality of redundant data has a plurality of state memory elements to store a plurality of redundant states; and the digital circuit further comprises:

means for copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

35. A digital circuit as in claim 31, wherein said means for processing the plurality of redundant results to maintain data integrity comprises means for detecting a computation error according to the redundant results.

36. A digital circuit as in claim 20, further comprising:

means for reprocessing, responsive to the computation error, the plurality of redundant data in a plurality of time slots in the time-multiplexed circuit of the digital circuit to generate a plurality of redundant results respectively.

37. A digital circuit as in claim 31, wherein the plurality of redundant results are processed to determine, in the plurality of time slots, a plurality of voting results.

38. A digital circuit as in claim 31, wherein said means for processing the plurality of redundant data comprises a combinatorial logic circuit.

39. A digital circuit as in claim 31, wherein said means for processing the plurality of redundant data pipelines processing of the plurality of redundant data.

40. A digital circuit as in claim 39, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for said means for processing the plurality of redundant data.

41. A digital circuit as in claim 40, wherein the plurality of redundant data is for a first channel; a plurality of redundant data for a second channel is assigned into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

42. A digital circuit as in claim 39, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for said means for processing the plurality of redundant data.

43. A digital circuit as in claim 42, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time

slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel.

44. A digital circuit as in claim 31, wherein said means for processing the plurality of redundant data circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

45. A digital circuit as in claim 31, wherein the digital circuit is integrated on one chip.

46. A method to design a digital circuit with redundancy protection, the method comprising:

automatically generating a second design of a time multiplexed circuit from a first design of a single-channel circuit, the time multiplexed circuit being configured to process a plurality of redundant data in a plurality of time slots to generate respectively a plurality of redundant results; and

generating a voting circuit to process the plurality of redundant results to maintain data integrity.

47. A method as in claim 46, wherein the voting circuit determines an output result according to a majority of the redundant results.

48. A method as in claim 47, wherein the voting circuit identifies a faulty one of the plurality of redundant results.

49. A method as in claim 48, further comprising:

generating a reload logic circuit, the time-multiplexed circuit having a plurality of state memory elements to store a plurality of redundant states, the reload logic circuit copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

50. A method as in claim 47, wherein the voting circuit detects a computation error according to the redundant results.

51. A method as in claim 50, further comprising:

generating a restart logic circuit coupled to the voting circuit and the time-multiplexed circuit, the restart logic circuit causing the time-multiplexed circuit to reprocessing the plurality of redundant data in response to the voting circuit detecting the computation error.

52. A method as in claim 46, wherein the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results.

53. A method as in claim 46, wherein the time-multiplexed circuit comprises a combinatorial logic circuit.

54. A method as in claim 46, further comprising:

generating a time multiplexer to assign the plurality of redundant data into the plurality of time slots;

wherein the time-multiplexed circuit pipelines processing of the plurality of redundant data.

55. A method as in claim 54, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit.

56. A method as in claim 55, wherein the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

57. A method as in claim 54, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit.

58. A method as in claim 57, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel.

59. A method as in claim 46, wherein the time-multiplexed circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

60. A method as in claim 46, wherein said generating the second design comprises:

generating a multi-state Finite-State-Machine (FSM) to time multiplex access to logic elements of the first design by multiple channels according to time slots which comprise the plurality of time slots for the plurality of redundant data.

61. A method as in claim 60, wherein said generating the second design further comprises:

generating a multiplexing circuit to time multiplex multiple inputs for the multiple channels onto an input line of the first design, the multiple inputs comprising the plurality of redundant data.

62. A method as in claim 60, wherein said generating the second design further comprises:

replacing a channel-specific element in the first design with multiple corresponding elements, each of the multiple corresponding elements being accessed for one of the multiple channels according to a state of the FSM.

63. A method as in claim 62, wherein the channel-specific element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

64. A method as in claim 62, wherein the channel-specific element is a channel-specific sequential element.

65. A method as in claim 64, further comprising:

identifying non-channel-specific sequential elements.

66. A method as in claim 65, wherein the non-channel-specific sequential elements comprise a set of pipeline register.

67. A method as in claim 64, wherein the channel-specific sequential element is replaced with a cascade of multiple shifting sequential elements.

68. A method as in claim 64, wherein the channel-specific sequential element is replaced with multiple memory elements addressed according to the state of the FSM.

69. A method as in claim 64, further comprising:

determining a number of feed-forward cutsets of sequential elements as non-channel-specific sequential elements.

70. A method as in claim 46, wherein said generating the second design further comprises:

replacing a sequential element in the first design with corresponding elements to generate the second design, the corresponding elements

being sequentially accessed in the second design according to timing for processing signals from multiple channels.

71. A method as in claim 70, wherein the corresponding elements are addressed sequentially.

72. A method as in claim 70, wherein the sequential element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

73. A method as in claim 46, wherein said generating the second design comprises:

generating a conglomerate of single-channel circuits of the first design; and

applying a folding transformation to the conglomerate of single-channel circuits to generate the second design of the time multiplexed circuit.

74. A method as in claim 73, further comprising:

generating information indicating a parallelism in the conglomerate of single-channel circuits;

wherein the folding transformation uses the information indicating the parallelism.

75. A method as in claim 74, wherein the information comprises information of a folding set.

76. A machine readable medium containing executable computer program instructions which when executed by a digital processing system cause said system to perform a method to design a digital circuit with redundancy protection, the method comprising:

automatically generating a second design of a time multiplexed circuit from a first design of a single-channel circuit, the time multiplexed circuit being configured to process a plurality of redundant data in a plurality of time slots to generate respectively a plurality of redundant results; and

generating a voting circuit to process the plurality of redundant results to maintain data integrity.

77. A medium as in claim 76, wherein the voting circuit determines an output result according to a majority of the redundant results.

78. A medium as in claim 77, wherein the voting circuit identifies a faulty one of the plurality of redundant results.

79. A medium as in claim 78, wherein the method further comprises:

generating a reload logic circuit, the time-multiplexed circuit having a plurality of state memory elements to store a plurality of redundant states, the reload logic circuit copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

80. A medium as in claim 76, wherein the voting circuit detects a computation error according to the redundant results.

81. A medium as in claim 80, wherein the method further comprises:

generating a restart logic circuit coupled to the voting circuit and the time-multiplexed circuit, the restart logic circuit causing the time-multiplexed circuit to reprocessing the plurality of redundant data in response to the voting circuit detecting the computation error.

82. A medium as in claim 76, wherein the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results.

83. A medium as in claim 76, wherein the time-multiplexed circuit comprises a combinatorial logic circuit.

84. A medium as in claim 76, wherein the method further comprises:

generating a time multiplexer to assign the plurality of redundant data into the plurality of time slots;

wherein the time-multiplexed circuit pipelines processing of the plurality of redundant data.

85. A medium as in claim 84, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit.

86. A medium as in claim 85, wherein the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

87. A medium as in claim 84, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit.

88. A medium as in claim 87, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel.

89. A medium as in claim 76, wherein the time-multiplexed circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

90. A medium as in claim 76, wherein said generating the second design comprises:

generating a multi-state Finite-State-Machine (FSM) to time multiplex access to logic elements of the first design by multiple channels according to time slots which comprise the plurality of time slots for the plurality of redundant data.

91. A medium as in claim 90, wherein said generating the second design further comprises:

generating a multiplexing circuit to time multiplex multiple inputs for the multiple channels onto an input line of the first design, the multiple inputs comprising the plurality of redundant data.

92. A medium as in claim 90, wherein said generating the second design further comprises:

replacing a channel-specific element in the first design with multiple corresponding elements, each of the multiple corresponding elements being accessed for one of the multiple channels according to a state of the FSM.

93. A medium as in claim 92, wherein the channel-specific element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

94. A medium as in claim 92, wherein the channel-specific element is a channel-specific sequential element.

95. A medium as in claim 94, wherein the method further comprises:

identifying non-channel-specific sequential elements.

96. A medium as in claim 95, wherein the non-channel-specific sequential elements comprise a set of pipeline register.

97. A medium as in claim 94, wherein the channel-specific sequential element is replaced with a cascade of multiple shifting sequential elements.

98. A medium as in claim 94, wherein the channel-specific sequential element is replaced with multiple memory elements addressed according to the state of the FSM.

99. A medium as in claim 94, wherein the method further comprises:

determining a number of feed-forward cutsets of sequential elements as non-channel-specific sequential elements.

100. A medium as in claim 76, wherein said generating the second design further comprises:

replacing a sequential element in the first design with corresponding elements to generate the second design, the corresponding elements being sequentially accessed in the second design according to timing for processing signals from multiple channels.

101. A medium as in claim 100, wherein the corresponding elements are addressed sequentially.

102. A medium as in claim 100, wherein the sequential element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

103. A medium as in claim 76, wherein said generating the second design comprises:

generating a conglomerate of single-channel circuits of the first design; and

applying a folding transformation to the conglomerate of single-channel circuits to generate the second design of the time multiplexed circuit.

104. A medium as in claim 103, wherein the method further comprises:

generating information indicating a parallelism in the conglomerate of single-channel circuits;

wherein the folding transformation uses the information indicating the parallelism.

105. A medium as in claim 104, wherein the information comprises information of a folding set.

106. A data processing system to design a digital circuit with redundancy protection, the data processing system comprising:

means for automatically generating a second design of a time multiplexed circuit from a first design of a single-channel circuit, the time multiplexed circuit being configured to process a plurality of redundant data in a plurality of time slots to generate respectively a plurality of redundant results; and

means for generating a voting circuit to process the plurality of redundant results to maintain data integrity.

107. A data processing system as in claim 106, wherein the voting circuit determines an output result according to a majority of the redundant results.

108. A data processing system as in claim 107, wherein the voting circuit identifies a faulty one of the plurality of redundant results.

109. A data processing system as in claim 108, further comprising:

means for generating a reload logic circuit, the time-multiplexed circuit having a plurality of state memory elements to store a plurality of

redundant states, the reload logic circuit copying data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results.

110. A data processing system as in claim 106, wherein the voting circuit detects a computation error according to the redundant results.

111. A data processing system as in claim 110, further comprising:

means for generating a restart logic circuit coupled to the voting circuit and the time-multiplexed circuit, the restart logic circuit causing the time-multiplexed circuit to reprocessing the plurality of redundant data in response to the voting circuit detecting the computation error.

112. A data processing system as in claim 106, wherein the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results.

113. A data processing system as in claim 106, wherein the time-multiplexed circuit comprises a combinatorial logic circuit.

114. A data processing system as in claim 106, further comprising:

means for generating a time multiplexer to assign the plurality of redundant data into the plurality of time slots;

wherein the time-multiplexed circuit pipelines processing of the plurality of redundant data.

115. A data processing system as in claim 114, wherein the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit.

116. A data processing system as in claim 115, wherein the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel.

117. A data processing system as in claim 114, wherein the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit.

118. A data processing system as in claim 117, wherein the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel;

and, data for the first channel is independent from data for the second channel.

119. A data processing system as in claim 106, wherein the time-multiplexed circuit comprises a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots.

120. A data processing system as in claim 106, wherein said means for generating the second design comprises:

means for generating a multi-state Finite-State-Machine (FSM) to time multiplex access to logic elements of the first design by multiple channels according to time slots which comprise the plurality of time slots for the plurality of redundant data.

121. A data processing system as in claim 120, wherein said means for generating the second design further comprises:

means for generating a multiplexing circuit to time multiplex multiple inputs for the multiple channels onto an input line of the first design, the multiple inputs comprising the plurality of redundant data.

122. A data processing system as in claim 120, wherein said means for generating the second design further comprises:

means for replacing a channel-specific element in the first design with multiple corresponding elements, each of the multiple corresponding elements being accessed for one of the multiple channels according to a state of the FSM.

123. A data processing system as in claim 122, wherein the channel-specific element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

124. A data processing system as in claim 122, wherein the channel-specific element is a channel-specific sequential element.

125. A data processing system as in claim 124, further comprising:

means for identifying non-channel-specific sequential elements.

126. A data processing system as in claim 125, wherein the non-channel-specific sequential elements comprise a set of pipeline register.

127. A data processing system as in claim 124, wherein the channel-specific sequential element is replaced with a cascade of multiple shifting sequential elements.

128. A data processing system as in claim 124, wherein the channel-specific sequential element is replaced with multiple memory elements addressed according to the state of the FSM.

129. A data processing system as in claim 124, further comprising:
means for determining a number of feed-forward cutsets of sequential elements as non-channel-specific sequential elements.

130. A data processing system as in claim 106, wherein said means for generating the second design further comprises:
means for replacing a sequential element in the first design with corresponding elements to generate the second design, the corresponding elements being sequentially accessed in the second design according to timing for processing signals from multiple channels.

131. A data processing system as in claim 130, wherein the corresponding elements are addressed sequentially.

132. A data processing system as in claim 130, wherein the sequential element comprises one of:

a) a constant;

b) a Random Access Memory (RAM) element;

c) a Read Only Memory (ROM) element;

d) a register;

e) a flip-flop; and

f) a negative latency register.

133. A data processing system as in claim 106, wherein said means for generating the second design comprises:

means for generating a conglomerate of single-channel circuits of the first design; and

means for applying a folding transformation to the conglomerate of single-channel circuits to generate the second design of the time multiplexed circuit.

134. A data processing system as in claim 133, further comprising:

means for generating information indicating a parallelism in the conglomerate of single-channel circuits;

wherein the folding transformation uses the information indicating the parallelism.

135. A data processing system as in claim 134, wherein the information comprises information of a folding set.